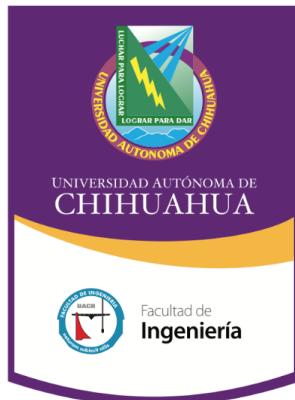


UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA
FACULTAD DE INGENIERÍA



CIRCUITOS LÓGICOS II
GRUPO: 7HW1

PRÁCTICA NO. 4
“DECODIFICADOR BCD”

EQUIPO
“*sudo rm -rf / --no-preserve-root*”

INTEGRANTES DEL EQUIPO
JESÚS ALEJANDRO JIMÉNEZ HERNÁNDEZ
OSCAR ALEJANDRO SERRANO PIZARRO
EMILIO ALBERTO SALAS RIOS

FECHA DE ENTREGA
DOMINGO 2 DE OCTUBRE DE 2022

DOCENTE
JESÚS MANUEL MUÑOZ LARGUERO

ÍNDICE

INTRODUCCIÓN	3
Objetivos	3
DESARROLLO	4
1. Configurar el Software Quartus II para que pueda funcionar con el CPLD MAX II.	4
2. Armar un circuito de salida que incluya las entradas y salidas necesarias para operar un circuito decodificador 7447 o 7448.	4
3. Compilar y grabar en el CPLD un programa que mediante un diagrama esquemático y código VHDL un código para nuestro decodificador.	5
CONCLUSIÓN	9
BIBLIOGRAFÍA	10

INTRODUCCIÓN

En esta práctica se nos pide configurar el CPLD como decodificador BCD para un display de siete segmentos de las dos fórmulas vistas anteriormente, con el diagrama lógico y código VHDL. Dependiendo del tipo de display, se necesitará hacer la configuración para el 7447 (ánodo) o 7448 (cátodo).

Objetivos

- Describir en lenguaje VHDL un código que sirva como decodificador BCD a 7 segmentos.
- Crear un diagrama esquemático con un decodificador de 7 segmentos 7447 o 7448.
- Armar el circuito y probar que funcione de distintas formas el decodificador BCD.

DESARROLLO

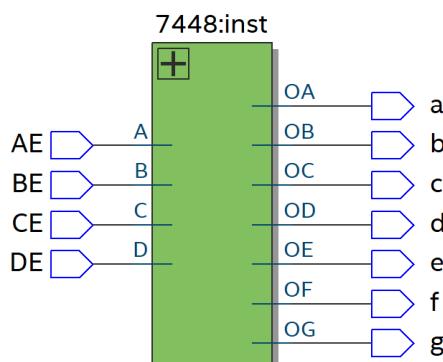
1. Configurar el Software Quartus II para que pueda funcionar con el CPLD MAX II.

La configuración del software está indicada dentro del documento que provee las instrucciones para la práctica, es por ello que tomamos la decisión de no incluirlo en el reporte de esta práctica. El documento puede ser encontrado en la bibliografía.

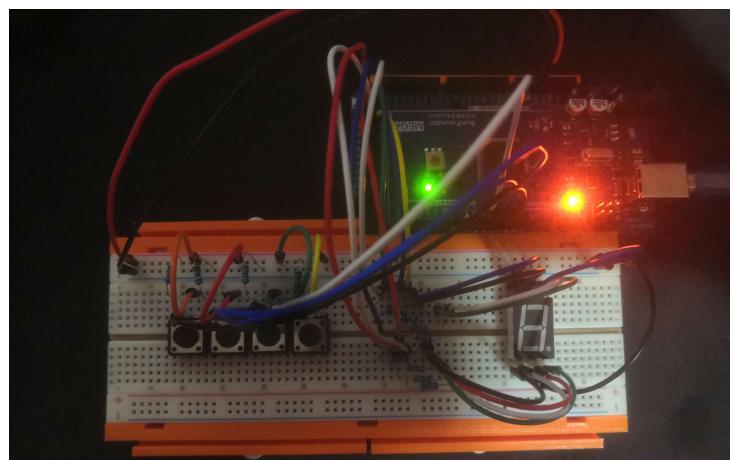
NOTA: Es importante aclarar que durante esta práctica, el CPLD presentó fallas que aún no logramos solucionar, por lo que no fue posible mostrar y grabar el circuito en funcionamiento, sin embargo se incluye y muestran imágenes del circuito físico así como la implementación del mismo a través de Arduino para mostrar lo que debería hacer el circuito.

2. Armar un circuito de salida que incluya las entradas y salidas necesarias para operar un circuito decodificador 7447 o 7448.

Este es el diagrama RTL del circuito físico, en el siguiente punto se muestra la configuración dentro del programa, esta imagen es únicamente didáctica.

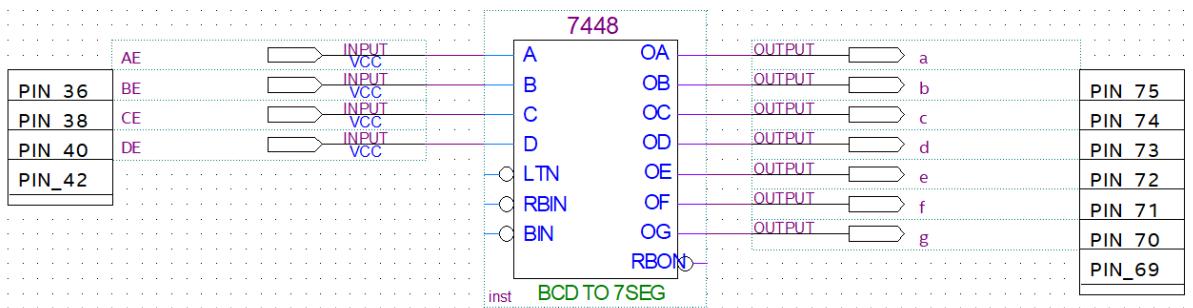


Y finalmente, se muestra el circuito físico: https://youtu.be/gAA_HHOhQ20



3. Compilar y grabar en el CPLD un programa que mediante un diagrama esquemático y código VHDL un código para nuestro decodificador.

En nuestro caso, se utilizó un display de siete segmentos de cátodo común, por lo que el decodificador que utilizamos para el circuito es el 7448.



Como podemos observar en esta imagen, se tiene el diagrama lógico del funcionamiento del programa que se instalará en el CPLD.

Por su parte, el código de VHDL implementado es el siguiente:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Practica_4_Parte2 is
Port (
    AE : in STD_LOGIC;
    BE : in STD_LOGIC;
    CE : in STD_LOGIC;
    DE : in STD_LOGIC;
    a : out STD_LOGIC;
    b : out STD_LOGIC;
    c : out STD_LOGIC;
    d : out STD_LOGIC;
    e : out STD_LOGIC;
    f : out STD_LOGIC;
    g : out STD_LOGIC
);
end Practica_4_Parte2;

architecture Behavioral of Practica_4_Parte2 is
begin
    a <= AE OR CE OR (BE AND DE) OR (NOT BE AND NOT DE);
    b <= (NOT DE) OR (NOT CE AND NOT DE) OR (CE AND DE);
    c <= BE OR NOT CE OR DE;
    d <= (NOT BE AND NOT CE) OR (CE AND NOT DE) OR (BE AND NOT CE AND DE) OR (NOT BE AND CE) OR AE;
    e <= (NOT BE AND NOT DE) OR (CE AND NOT DE);
    f <= AE OR (NOT CE AND NOT DE) OR (BE AND NOT CE) OR (BE AND NOT DE);
    g <= AE OR (BE AND NOT CE) OR (NOT BE AND CE) OR (CE AND NOT DE);
end Behavioral;

```

Durante la investigación para resolver esta práctica, también encontramos una segunda forma de poder implementarlo a partir de código VHDL, sin embargo, debido a que no hemos visto este tipo de implementación en clase, únicamente queremos incluirlo como complemento. A su vez, no nos fue posible comprobar el éxito de los códigos debido a la falla en el CPLD anteriormente mencionada. Más adelante se incluye el diagrama RTL del código que comprueba su “funcionamiento teórico”.

VHDL Program for BCD to Seven Segment Decoder:

```
1. library ieee;
2. use ieee.std_logic_1164.all;
3.
4. entity seg7 is
5.   port (bcd : in std_logic_vector(3 downto 0);
6.         leds : out std_logic_vector(1 to 7));
7. end seg7;
8.
9. architecture behaviour of seg7 is
10. begin
11.   process (bcd)
12.   begin
13.     case bcd is
14.       when "0000" => leds <= "1111110";
15.       when "0001" => leds <= "0110000";
16.       when "0010" => leds <= "1101101";
17.       when "0011" => leds <= "1111001";
18.       when "0100" => leds <= "0110011";
19.       when "0101" => leds <= "1011011";
20.       when "0110" => leds <= "1011111";
21.       when "0111" => leds <= "1110000";
22.       when "1000" => leds <= "1111111";
23.       when "1001" => leds <= "1110011";
24.       when others => leds <= "-----";
25.     end case;
26.   end process;
27. end behaviour;
```

Debido a que este código no es nuestro, en la bibliografía tenemos un link directo al artículo que lo contenía.

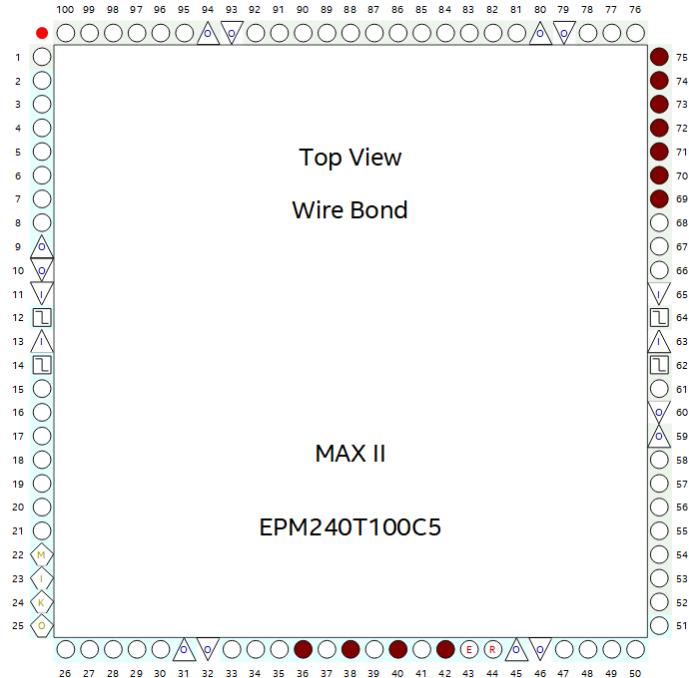
Continuando, la configuración de entrada es la siguiente:

- Pin 36 → Entrada A → Botón AE
- Pin 38 → Entrada B → Botón BE
- Pin 40 → Entrada C → Botón CE
- Pin 42 → Entrada D → Botón DE

Las compuertas de salida se encuentran de la siguiente forma:

- Pin 75 → Salida a → Segmento 'a' del Display de 7 Segmentos
- Pin 74 → Salida b → Segmento 'b' del Display de 7 Segmentos
- Pin 73 → Salida c → Segmento 'c' del Display de 7 Segmentos
- Pin 72 → Salida d → Segmento 'd' del Display de 7 Segmentos
- Pin 71 → Salida e → Segmento 'e' del Display de 7 Segmentos
- Pin 70 → Salida f → Segmento 'f' del Display de 7 Segmentos
- Pin 69 → Salida g → Segmento 'g' del Display de 7 Segmentos

Es importante aclarar que esta configuración aplica para ambos, el código VHDL y el diagrama lógico. En esta imagen se muestra la configuración de pines necesaria para programar el CPLD.



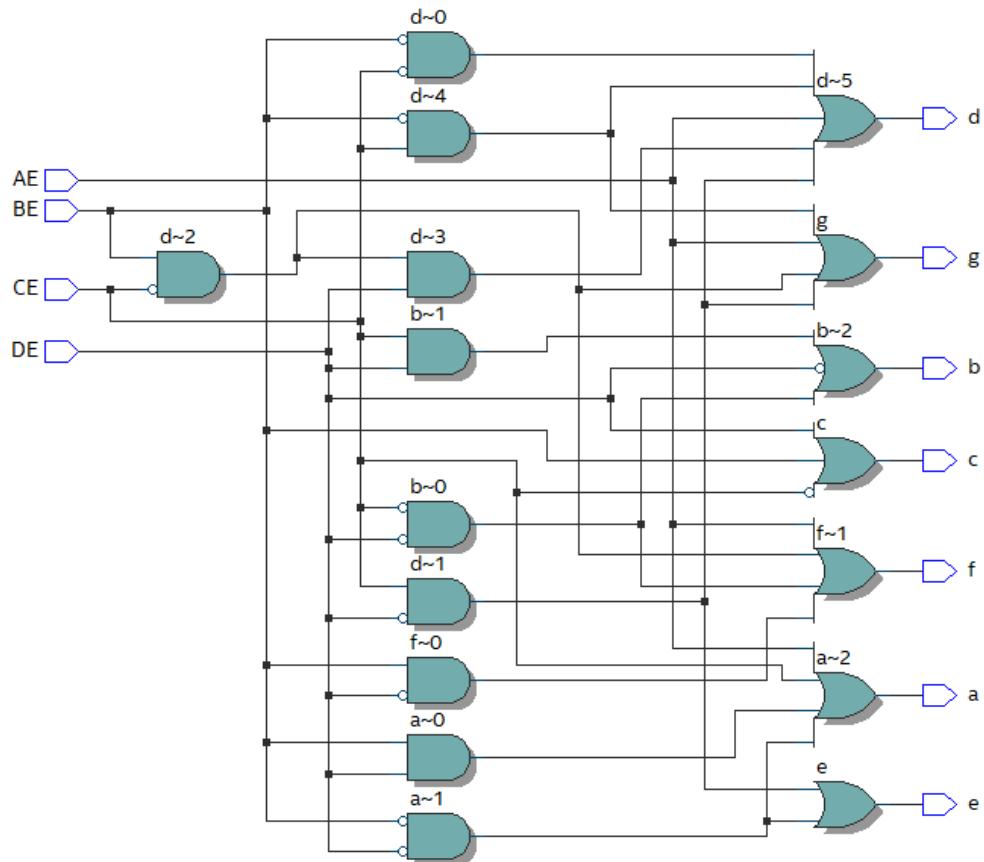
Esta imagen nos muestra los pines de entrada anteriormente mencionados.

Node Name	Direction	Location	I/O Bank	I/O Standard
in AE	Input	PIN_36	1	3.3-V L..efault)
in BE	Input	PIN_38	1	3.3-V L..efault)
in CE	Input	PIN_40	1	3.3-V L..efault)
in DE	Input	PIN_42	1	3.3-V L..efault)

Y finalmente, aquí tenemos la configuración de pines de salida del dispositivo.

Node Name	Direction	Location	I/O Bank	I/O Standard
out a	Output	PIN_75	2	3.3-V L..efault)
out b	Output	PIN_74	2	3.3-V L..efault)
out c	Output	PIN_73	2	3.3-V L..efault)
out d	Output	PIN_72	2	3.3-V L..efault)
out e	Output	PIN_71	2	3.3-V L..efault)
out f	Output	PIN_70	2	3.3-V L..efault)
out g	Output	PIN_69	2	3.3-V L..efault)

Para terminar, se muestra la captura del diagrama RTL del código VHDL:



Y la tabla de verdad del circuito:

ENTRADA

BIT 3	BIT 2	BIT 1	BIT 0	SALIDA							DECIMAL
A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

CONCLUSIÓN

En esta práctica se aplicó todos los conocimientos adquiridos hasta el momento. Implementamos todas las herramientas como el diagrama RTL para comprobar que tanto nuestro circuito lógico como nuestra implementación en código VHDL coincidieran. Y si bien no tuvimos la oportunidad de implementarlo físicamente, a través del arduino logramos observar que el circuito físico cumplía con lo solicitado, decodificar una señal de entrada a un display de siete segmentos. Analizando la tabla de verdad, podemos ver que el decodificador lo único que hace es pasar un byte o bien los valores de cuatro bits, a nuestro sistema decimal para que nosotros entendamos el número. Algo más a aclarar es que en el caso del diagrama lógico, al usar el 7448, no se puede apreciar todas las compuertas lógicas que comprenden al mismo como en el caso del código VHDL, pero al observar el datasheet, podemos ver que el diagrama es el mismo.

BIBLIOGRAFÍA

Documentación de práctica:

[https://aulas3.uach.mx/pluginfile.php/1115980/mod_assign/introattachment/0/
Practica%20%234%20-%20Decodificador.pdf?forcedownload=1](https://aulas3.uach.mx/pluginfile.php/1115980/mod_assign/introattachment/0/Practica%20%234%20-%20Decodificador.pdf?forcedownload=1)

Artículos adicionales:

<https://allaboutfpga.com/bcd-to-7-segment-decoder-vhdl-code/>

<https://buzztech.in/bcd-to-seven-segment-decoder-program-in-vhdl/>